GENERAL ILLUMINATION CORRECTION AND ITS APPLICATION TO FACE NORMALIZATION

Juhua Zhu, Bede Liu, Stuart C. Schwartz

Dept. of Electrical Engineering, Princeton University
E-Quad, Olden St., Princeton 08544, U.S.A.
E-mail: {juhuazhu, liu, stuart}@ee.princeton.edu

ABSTRACT

The appearance of an object can be severely affected by illumination. Thus, illumination correction is necessary both for human perception and machine recognition. This paper reports on a general approach for fast illumination correction. The approach has been tested for application in face normalization as a preprocessing step in face recognition. The basic idea of the algorithm is to locally normalize the image contrast using an affine transformation lighting model based on local estimation of background and gain. The background is estimated via an efficient multi-resolution low-pass filter and the gain is estimated via homomorphic filtering. This is followed by normalizing the data with the help of a clipped histogram. Experiments on images with different lighting conditions produce results that are better than those from using several popular illumination correction methods.

1. INTRODUCTION

For many vision algorithms, an illumination change in a scene such as a shadow will typically cause the algorithm to perform poorly. This usually occurs because the algorithm cannot distinguish by pixel value alone between the effect due to background light and that due to foreground object, thereby treating the lighting effect as part of the object.

Many well known enhancement algorithms such as histogram equalization are global in nature and are intended to view an image more clearly as a whole. They produce results that may not be satisfactory over some local regions. More effective approaches for enhancement are performed on a local neighborhood of each pixel rather than relying on an operation determined by the results of a global operation. Adaptive Histogram Equalization (AHE) [1] computes the histogram of a local window centered at a given pixel to determine the mapping for that pixel, which provides a local contrast enhancement. However, it often leads to noise amplification in "flat" regions of the image and "ring" artifacts at strong edges. It is also computationally intensive. Similar problems occur with Local Range Modification (LRM) [2], although it works faster. In addition, the LRM result is apparently blocky. LRM finds interpolated minimum and maximum pixel values in the context region and stretches them to the desired range via the equation

$$Y(x,y) = \frac{C}{(\max - \min)}(X(x,y) - \min) \qquad (1)$$

where the constant C depends on the data storage or display device.

Our objective for the algorithm in this paper is to take an image and process it while retaining a natural look so that any planned subsequent algorithm such as object detection or matching is still likely to work well under a lighting change.

2. GENERAL ILLUMINATION CORRECTION

2.1. Lighting Model

A traditional method of removing or reducing an unwanted shadow effect in an image is to subtract off a low-passed version of the image with a large kernel size. This serves to remove the slowly varying lighting condition, which is often associated with shadow or background lighting. However, a lighting change in an image often cannot be modeled well as a simple additive effect. Often what happens in practice is the range of dark to bright in an image gets amplified as the illumination increases, much like someone increasing the gain of the camera. This is a multiplicative effect in addition to the additive one. This also is often a local change rather than a global one, so that gain controls in the camera are not enough to compensate for the lighting change. In our algorithm, we attempt to adjust and to compensate to some degree for this type of effect on an image.

Mathematically, we model the lighting change as a local affine transformation of the pixel value.

$$G'(x,y) = A(x,y) \cdot G(x,y) + B(x,y) \qquad (2)$$

where $G(x,y)$ is the original image and $G'(x,y)$ is the modified image as a result of the local lighting. This means the lighting change has both an additive and a multiplicative effect on the result. The proposed normalization algorithm attempts to remove these affects by first subtracting off a low-passed estimate $\hat{B}(x,y)$ and then normalizing with respect to a gain estimate $\hat{A}(x,y)$. Explicitly, this means:

$$\hat{G}(x,y) = \frac{\kappa}{\hat{A}(x,y)} \cdot (G'(x,y) - \hat{B}(x,y)) \qquad (3)$$

If the estimates $\hat{A}(x,y)$ and $\hat{B}(x,y)$ are perfectly correct, then this algorithm will reconstruct the original image. However, since they must be estimated from the image itself, processing the modified image, $G'(x,y)$, is clearly not able to replicate the original image.

In our current algorithm, we have chosen to estimate the additive effect $B(x,y)$ by a multiresolution-based low-pass filter, which approximates a large kernel low-pass filter. We deal with the multiplicative effect $A(x,y)$ by means of homomorphic filtering. After obtaining these two estimates, the clipped histogram is used to help normalize the image.

2.2. Background Estimation

The low-pass filter needed to estimate the background lighting should have a very large kernel size. This is to eliminate the DC and typical shadow in later processing. However, this can be rather computationally time consuming: a separable $n \times n$ Gaussian blur would cost over $2n$ multiplications per pixel. For $n = 64$ or more, this could take much too long for practical purposes. Instead we present a fast approximate low-pass filter based on multi-resolution. The idea is fairly simple: a coarse estimate of a low-pass filter would be to take non-overlapping block average in the image which provides a low resolution image representation. A simple rescaling of this image to the original resolution would display very apparent blockiness. We instead "rescale" the low resolution image in stages, each intermediary resolution approaching step by step closer to the original resolution. One enlargement step goes though steps as is shown in figure 1. What this does is to create a multi-resolution spline interpolation of the reduced image.

In our actual implementation, each intermediary resolution is a size doubling and we choose the low-pass kernel of a separable 5×5 for each resolution. If the reduced image was obtained from the original one using a neighborhood size of $2^n \times 2^n$, the average number of multiplications per original image pixel is approximately:

$$10 \sum_{k=1}^{n} \frac{1}{4^k} \qquad (4)$$

If the neighborhood size is 32 ($n = 5$), we have on average about 3.3 multiplications per original image pixel. However, the 5×5 kernel on the lowest scale corresponds approximately to 160×160 kernel size in the original scale. A roughly equivalent separable low-pass filter would take 320 multiplications! So even though the multiresolution low-pass filter is only approximate, the savings in multiplications is considerable.

Since each stage doubles the image width and height in our implementation in a particular way ($2 \times height - 1$ and $2 \times width - 1$), it is rather unlikely that repeated operations on the reduced image will return to an image the same pixel width and height as the original. We simply overshoot the image size and then resize the overshot image to the appropriate size using nearest neighbor resampling.

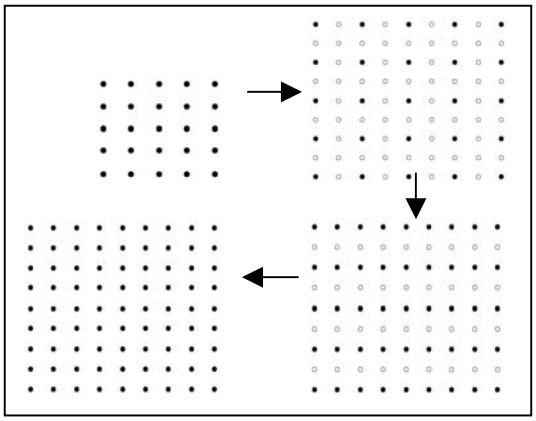

Fig.1. One step in the enlargement

2.3. Gain Estimation

We now turn to estimating the gain $A(x,y)$ and remove it. If we change the model (2) into the following form:

$$G'(x,y) - B(x,y) = A(x,y) \cdot G(x,y) \qquad (5)$$

Since the disturbance is multiplicative, we are naturally led to homomorphic filtering. $A(x,y)$ is expected to vary slowly across the image, while the image $G(x,y)$ usually varies rapidly. As suggested by homomorphic filtering, estimating the gain can be accomplished in the log domain.

We define:

$$f(x,y) = \ln(G'(x,y) - B(x,y)) = \ln A(x,y) + \ln G(x,y) \quad (6)$$

Then low pass filtering $f(x,y)$ will isolate $\ln A(x,y)$.

$$LPF(f(x,y)) \approx \ln A(x,y) \quad (7)$$

Finally, exponentiation is necessary to bring the quantity into the normal spatial domain.

$$A(x,y) = \exp(LPF(f(x,y))) \quad (8)$$

Traditionally, homomorphic filtering does a high-pass filtering in the frequency domain so as to preserve the high frequency part and to suppress the low frequency part, $G(x,y)$ and $A(x,y)$, respectively, in our case. When it deals with large images, the implementation becomes quite slow.

Recall we have a very fast multiresolution low-pass filter for background estimation. Similarly, it can be used here to estimate the gain. We do not simply subtract $\ln A(x,y)$ from $f(x,y)$ to get $\ln G(x,y)$, because we want to improve the estimate of $A(x,y)$ as described below.

2.4. Image Normalization

Up to this point, we may simply substitute all estimates into (3) to recover the image, with the additional step of rescaling the data to the allowable range of values for the storage device, [0, 255] for most computers. Since this step can move any data to any range, the selection of κ is not crucial. Anything making the data some reasonable quantity can be used.

However, we observed that there are often some noisy pixels located at the two extremes in the histogram. These pixels are sparse and do not represent the important features of the image. But they impede the image to make full use of the dynamic display range. In order to avoid reducing contrast, we propose to establish high and low thresholds based on the frequency of the pixel values. In the histogram, the gray level is searched from two ends, until the first gray level with high histogram frequency is found. The respective gray levels are set to T_{lo} and T_{hi}. Mathematically, we have

$$\hat{G}'(x,y) = \begin{cases} g_{lo}, & \hat{G}(x,y) \le T_{lo} \\ g_{hi}, & \hat{G}(x,y) \ge T_{hi} \\ \dfrac{g_{hi} - g_{lo}}{T_{hi} - T_{lo}}(\hat{G}(x,y) - T_{lo}) + g_{lo}, & else \end{cases} \quad (9)$$

where g_{lo} and g_{hi} are the lower and upper limit of storage medium, respectively.

2.5. Discussion

Since this is a general purpose algorithm, comparisons and performance will largely be a subjective matter. Generally speaking, our goal is to have a more natural look and more readable details than the original image.

Implicit in the algorithm are the neighborhood sizes of two low-pass filters, one for background estimation and the other for gain estimation. Clearly, large neighborhood sizes lead to smooth results, but the illumination effect may not be corrected as much as needed, while small sizes probably will generate some discontinuities of lighting effect. In our experiments, when both neighborhood sizes are set as $MIN(0.2 \times shortedge, 32)$, good results can be achieved for most images under slowly varying lighting conditions. However, for those images with too strong a contrast, which is often caused by light incidence with the angle of nearly 180°, it is better for us to use some smaller neighborhood sizes to balance the illumination.

In addition, as may be noticed, during the correction process, we keep subtracting low frequency components off from the original image. This will eventually make the high frequency prominent. In addition, in some cases when the original image has noise or has abrupt local lighting change, the gain does not necessarily vary very slowly. One easy way to compensate for this is to modify the gain estimate as:

$$A'(x,y) = A(x,y) \cdot m(G'(x,y)) \quad (10)$$

where $A(x,y)$ is the result of former gain estimation, and $m(\cdot)$ is some modification function for compensation. In our experiments, we set it as:

$$m(x) = \sqrt[4]{x} \quad (11)$$

What this modification does is to add some high frequency components to the gain estimate. As a result, the corrected image will have less high frequency components. The fourth root is employed based on experimental results. So, finally, we have the illumination correction function as:

$$\hat{G}(x,y) = N\left(\frac{\kappa}{\hat{A}(x,y) \cdot \sqrt[4]{G'(x,y)}}(G'(x,y) - \hat{B}(x,y))\right) \quad (12)$$

where $N(\cdot)$ means data normalization with the help of histogram clipping.

A better way to deal with different lighting conditions, slow variations or sharp changes, is to estimate the properties of the image and then automatically choose parameters. The difficulty with this approach is that it is often difficult to identify from the image itself if the change of grayscale value is due to the lighting effects or color effects.

3. ILLUMINATION IN FACE RECOGNITION

As stated by Moses et al. [3], "the variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity". Different illumination condition is a large impediment for face recognition.

The above method is we think a good choice to normalize face data. However, a slight change of the algorithm should be made to better serve the goal of face recognition. In face recognition, in addition to making it easier to recognize each face, it is also necessary to make the overall illumination of each face image to be the same for purposes of comparison. Thus, instead of rescaling the data to the range of [0, 255] for the benefit of the viewer, here, we need to normalize the data for the benefit of machine recognition. One alternative is to normalize the data by a certain mean value and standard deviation.

Experiments show that our method of illumination correction can achieve satisfactory results in removing annoying effects of shadowing and varied lighting for most face images. We compared the results with histogram equalization and histogram matching with the "perfect" histogram of the face [4].

However, since no geometric face-model is used, our method cannot recover those face signals with too many saturated points in the original images. Neither is it able to do a good job for those faces with very strong shadows.

4. EXPERIMENTAL RESULTS

We have evaluated the proposed method on images with different lighting conditions and different objects. We compared the results with those of Histogram Equalization (HE), Adaptive HE, Histogram Matching and Local Range Modification. Based on our preliminary experiments, the results indicate better performance in visualization while still maintaining fast computation.

Fig.2. shows the result of the proposed algorithm on an image with slowly varying illumination. Fig.3. includes three groups of face images, each corresponding to different lighting variability. The first row has one lighting source from the right, the second has one stronger source from left, and the last has one weak source from left. It is seen that the processed faces by our method tend to be more homogeneous and normalized than the other results, even for the image with a patch of saturated points. That will definitely help sample clustering and pattern classification.

However, for very extreme illumination (like some face samples in the Harvard database), the algorithm does not appear to correct very much for illumination.

For more detailed results, please refer to:
http://www.ee.princeton.edu/~juhuazhu/Acad/illum.htm

6. REFERENCES

[1] S.M. Pizer, E.P. Amburn, J.D. Austin, et. Al., "Adaptive Histogram Equalization and Its Variation," *Computer Vision, Graphics, and Image Processing,* Vol.39, pp. 355-368, 1987.

[2] J.D. Fahnestock, and R.A. Schowengerdt, "Spatially Variant Contrast Enhancement Using Local Range Modification," *Optical Engineering,* Vol.22, No.3, pp. 378-381, 1983.

[3] Y. Moses, Y. Adini, and S. Ullman, "Face Recognition: The Problem of Compensating for Changes in Illumination Direction," *European Conf. Computer Vision*, PP.286-296, 1994

[4] http://web.media.mit.edu/~jebara/uthesis/node59.html

Fig.2. (a)Original image, (b)HE result, (c)LRM result (neighborhood=32x32), and (d)result by proposed method (neighborhood=32x32). (Larger images online.)

Fig.3. Original images (from Purdue face database) with different lighting in left most column, HE results in the second, followed by Histogram Matching results, and the right most are results of the proposed algorithm.